

String Vector based AHC Variants for Clustering Words

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we propose and apply the string vector based AHC variants to the word clustering. The initial AHC version which receives a string vector, was previously proposed as a tool of the word clustering. In this research, we mention the three AHC variants: one where the data clustering proceeds in the bottom-up direction with the similarity threshold, one where it allows any merge of more than two pairs, and one where clusters are merged based on the radius. In this research, we modify the three AHC variants into the string vector-based versions, as well as the initial AHC version. As the goal of this research, we improve the clustering performance by modifying them so.

I. INTRODUCTION

The word clustering refers to the process of segmenting a single group of words into subgroups, each of which includes similar words. It is possible to cluster words by their prefixes, postfixes, and infixes, and this kind of word clustering is called lexical word clustering or syntactical word clustering. The kind of word clustering which is covered in this research is called semantic word clustering where words are clustered by their meanings; the lexical word clustering is excluded from this research. The semantic word clustering belongs to the hard clustering where each word is included into only one category; it will be expanded into the soft word clustering or the hierarchical word clustering in future research. The clusters as results from clustering words are viewed as topics, in this research.

This research is motivated by the successful results from applying the string vector based AHC algorithm to the word clustering. A string vector as a word representation is a finite set of strings with their own meanings, and the similarity metric between two string vectors was defined. The AHC algorithm was modified into the string vector-based version which processes string vectors directly and applied to the word clustering. Its better performance was empirically validated, compared with the traditional AHC algorithm. We derive some AHC variants and modify them into the string vector-based versions as the approaches to the word clustering.

The idea of this research is to modify the three AHC variants into the versions each of which takes a string vector as its input, as the approaches to the word clustering. In the previous work, the initial AHC algorithm was modified

into the string vector-based version for upgrading the word clustering performance, compared with the traditional AHC algorithm. We mention the three AHC variants: one where clustering data items proceeds in the bottom up direction depending on the similarity threshold, instead of the desirable number of clusters, one where merging more than one pair is allowed in each iteration, and one where a cluster is merged with others within its radius. In this research, the three AHC variants are modified into the string vector-based versions by the semantic operation on the string vectors and applied to the semantic word clustering. The performances of AHC variants are improved as the approaches to the word clustering.

Let us mention some benefits from this research. The problems such as the huge dimensionality and the sparse distribution from encoding words into numerical vectors are solved by encoding them into string vectors. The AHC algorithm is replaced by its variants for improving the clustering speeds. The AHC variants are modified into their string vector-based versions for improving the clustering performances. The goal of this research is to provide better approaches to the word clustering with respect to both the clustering speed and the clustering performance.

Let us mention the organization of this paper. In Section II, we explore the previous works which are relevant to this study. In Section III, we describe in detail what is proposed in this research. In Section IV, we mention the significances of this research and the remaining tasks. This paper is composed of the four sections.

II. PREVIOUS WORKS

This section is concerned with the previous works which are relevant to this research. It is intended to expand the style of modifying the AHC algorithm to its three variants. The directions of exploring previous works are derivation of AHC variants, modification of the standard AHC algorithm into its string vector-based version, and the cases of encoding a text into a string vector. The distinguishable points of this research are derivation of three AHC variants from the standard version, modification of them into string vector-based versions, and their applications to the word clustering. This section is intended to explore previous works for providing the background for this research.

Let us explore the previous cases of deriving AHC variants. In 2012, Murtagh and Contreras mentioned the possibility of deriving various versions of AHC algorithm and a particular variant which uses nearest neighbors [3]. In 2013, Sasirekha and Baby presented various similarity metrics between vectors and strategies of defining a similarity between clusters [4]. In 2015, Nazari et al. proposed the AHC algorithm which merges two clusters based on their common nearest neighbors [6]. In the AHC algorithms which are mentioned in above previous literatures, only one pair of clusters is merged in each iteration.

Let us explore the previous works which are concerned with the application of the modified version of the standard AHC algorithm to word clustering. In 2016, Jo initially proposed that the string vector-based version should be applied to the word clustering [7]. In 2018, Jo validated empirically its performances in the word clustering [9]. In 2023, he prepares the journal article which describes the string vector-based AHC algorithm and its application to the word clustering for its publication [12]. What is provided by the previous works becomes the basis for this research.

Let us explore the previous works on the neural networks, NTC (Neural Text Categorizer), where encoding a text into a string vector was initially proposed. In 2010, it was initially invented as an approach to the text classification by Jo [1]. In 2015, it was applied to the topic identification of Arabic texts by Abainia [5]. In 2018, it was evaluated as the most practical neural networks by Lakshmi and Rao [10]. In 2010, the NTSO (Neural Text Self Organizer) was also invented as an approach to the text clustering by Jo [2].

Let us mention the differences of this research from the previous works which are explored above. In this research, we derive the AHC variants as fast clustering algorithms by allowing merging multiple cluster pairs for each iteration. The three AHC variants are modified into string vector-based versions and adopted for implementing the word clustering system. In this research, the string vector based AHC variants are developed as clustering algorithms as well as classification algorithms. We will describe the modified AHC variants in Section III.

III. PROPOSED WORK

This section is concerned with what is proposed in this research. In Section III-A, we describe the process of encoding a word into a string vector and the proposed similarity metric. In Section III-B, we describe the standard version of AHC algorithm where the proposed similarity metric is adopted. In Section III-C, we derive the three variants from the standard version. In Section III-D, we present the system architecture of the word clustering system.

A. Similarity Metric

This section is concerned with the string vector as the representation of a word and the similarity between string

vectors. The string vector is defined as a finite ordered set of strings; in the vector, the numerical values are replaced by the strings. It is required to define a similarity matrix for computing the similarity between elements. The similarity between string vectors is the average over one-to-one similarities between their elements. This section is intended to describe the process of encoding a word into a string vector and the process of computing the similarity between string vectors.

The process of encoding words into string vectors is illustrated in Figure 1. In the word-text association, a text collection is constructed by gathering texts, and each word is associated with its own texts based on their information. The string vector features are defined as symbolic forms manually in the feature definition. In the feature value assignment, the string vector which represents a word is filled with text identifiers which correspond to the features. Refer to [8] for studying the encoding process in more detail.

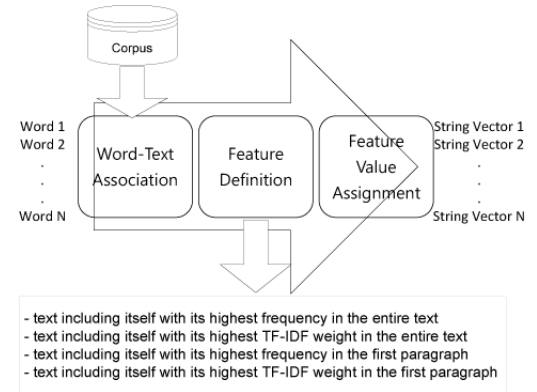


Figure 1. Process of Encoding Words into String Vectors

The similarity matrix which is the basis for computing the similarity between string vectors is illustrated in figure 00. In the matrix, each row and each column correspond to text identifiers, and each element is the similarity between texts, d_i and d_j , as shown in equation (1),

$$s_{ij} = \text{sim}(d_i, d_j) \quad (1)$$

The similarity between two texts, d_i and d_j , is computed by equation (2),

$$\text{sim}(d_i, d_j) = \frac{2|D_i \cap D_j|}{|D_i| + |D_j|} \quad (2)$$

where D_i is the set of words in the text, d_i , and D_j is the set of words in the text, d_j . The value of similarity between texts, d_i and d_j , is always given as a normalized value between zero and one, as shown in equation (3)

$$0 \leq \text{sim}(d_i, d_j) \leq 1 \quad (3)$$

The similarity matrix is used for computing the similarity between string vectors which represent words as a reference table.

$$\begin{bmatrix} s_{11} & s_{12} & \dots & s_{1N} \\ s_{21} & s_{22} & \dots & s_{2N} \\ \dots & \dots & \dots & \dots \\ s_{N1} & s_{N2} & \dots & s_{NN} \end{bmatrix} \quad \text{sim}(d_i, d_j) = \frac{2|D_i \cap D_j|}{|D_i| + |D_j|}$$

$$0 \leq \text{sim}(d_i, d_j) \leq 1.0$$

Figure 2. Semantic Similarity Matrix

Let us mention the process of computing the similarity between string vectors as a semantic similarity between words. The two string vectors which represent words are notated by $\text{str}_1 = [d_{11} \ d_{12} \ \dots \ d_{1d}]$ and $\text{str}_2 = [d_{21} \ d_{22} \ \dots \ d_{2d}]$. The similarity between two string vectors is computed by equation (4),

$$\text{sim}(\text{str}_1, \text{str}_2) = \frac{1}{d} \sum_{i=1}^d \text{sim}(d_{1i}, d_{2i}). \quad (4)$$

The similarity between elements, $\text{sim}(d_{i1}, d_{2i})$, is taken from the similarity which was mentioned above. The value which is generated by equation (4) is always given as a normalized value between zero and one as shown in equation (5),

$$0 \leq \text{sim}(\text{str}_1, \text{str}_2) \leq 1. \quad (5)$$

Let us make some remarks on the process of encoding words into string vectors and computing the similarity between words. A word is encoded into a string vector by the word-text association, the feature definition, and the feature value assignment. The similarity matrix is defined as the basis for computing the similarity between string vectors. The similarity is computed by equation (4). In the future research, we consider representing a word into multiple string vectors.

B. Initial Version of String Vector based AHC Algorithm

This section is concerned with the initial version of string vector based AHC algorithm. The version of AHC algorithm was initially proposed as the approach to the text clustering by Jo in 2019 [11]. The similarity metric between string vectors which was described in the previous section is used for computing the similarity between clusters. The AHC algorithm proceeds clustering data items with the bottom-up direction. This section is intended to describe the initial version of AHC algorithm from which its variants are derived.

The initial status for applying the AHC algorithm to the word clustering is illustrated in Figure 3. The words as clustering targets are encoded into string vectors by the process, which was described in Section III-A, and they are notated by a set, $Tr = \{\text{str}_1, \text{str}_2, \dots, \text{str}_N\}$. The clusters are defined as many as data items, as $C_1, C_2, \dots, C_N$, and each cluster includes a data item as $C_i = \{\text{str}_i\}$. The initial status which is represented in Figure 00 is notated by

$C_1 = \{\text{str}_1\}, C_2 = \{\text{str}_2\}, \dots, C_N = \{\text{str}_N\}$. The cluster with only one item is called singleton, and singletons are constructed as many as data items in the initial status.



Figure 3. Initial Clusters in using AHC Algorithm: Singletons

The process of computing the similarity between two clusters is illustrated in Figure 4. The two clusters are notated by two sets of items, $C_1 = \{\text{str}_{11}, \text{str}_{12}, \dots, \text{str}_{1|C_1|}\}$ and $C_2 = \{\text{str}_{21}, \text{str}_{22}, \dots, \text{str}_{2|C_2|}\}$. The similarity between clusters, C_1 and C_2 , is computed by equation (6),

$$\text{sim}(C_1, C_2) = \frac{1}{|C_1| \cdot |C_2|} \sum_{i=1}^{|C_1|} \sum_{j=1}^{|C_2|} \text{sim}(\text{str}_{1i}, \text{str}_{2j}). \quad (6)$$

If the similarity metric which was described in Section III-A is adopted, the similarity between two clusters is always a normalized value between zero and one, as expressed in equation (7),

$$0 \leq \text{sim}(C_1, C_2) \leq 1. \quad (7)$$

The complexity of computing the similarity between two clusters is expressed by $O(|C_1||C_2|)$.

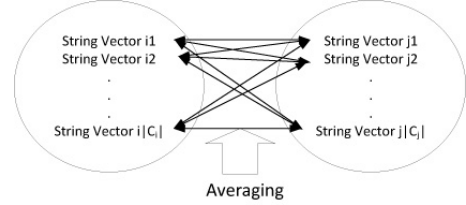


Figure 4. Similarity between Clusters

The process of clustering data items is illustrated as pseudo codes in Figure 5. The AHC algorithm begins with the status where singletons are given as many as data items. All possible pairs of clusters are generated from the current cluster list, their similarities are computed by equation (6), and the cluster pair with the highest similarity is merged into a cluster. In the AHC algorithm, the data items are clustered by iterating computing similarities of all possible cluster pairs and merge a cluster pair into a cluster. In each iteration, one cluster is decreased, and this iteration continues until reaching the desirable number of clusters.

Let us make some remarks on the initial version of the AHC algorithm from which we derive the variants. The initial status in applying the AHC algorithm to data clustering is that there are singletons as many as data items. The similarity between clusters is computed by averaging the similarities of all possible pairs from them. Data items are clustered by iterating computing the similarities of all

```

List clusterDataNumericalVectorList(List<String> vectorList, int desiredClusterSize){
    int itemSize = stringVectorList.size();

    if(itemSize <= desiredClusterNumber)
        return null;
    List<Cluster> clusterList = new List<>();
    //Initialize Clusters
    for(int i = 0; i < itemSize; i++){
        Cluster cc = new Cluster();
        StringVector dataItem = stringVectorList.getElement(i);
        cc.addDataItem(dataItem);
        clusterList.addElement(cc);
    }

    int clusterListSize = clusterList.size();

    //Cluster Data items in the bottom up direction
    while(clusterListSize > desiredClusterListSize){
        double maxSimilarity = 0.0;
        int maxIndex1 = 0;
        int maxIndex2 = 0;
        for(int i = 0; i < clusterListSize; i++){
            Cluster c1 = clusterList.getElement(i);
            for(int j = 0; j < clusterListSize; j++){
                Cluster c2 = clusterList.getElement(j);
                double similarity = sim(c1, c2);
                if(similarity > maxSimilarity){
                    maxSimilarity = similarity;
                    maxIndex1 = i;
                    maxIndex2 = j;
                }
            }
        }
        Cluster cmax1 = clusterList.getElement(maxIndex1);
        Cluster cmax2 = clusterList.getElement(maxIndex2);
        Cluster mergeCluster = cmax1.merge(cmax2);
        clusterList.deleteElement(cmax1);
        clusterList.deleteElement(cmax2);
        clusterList.addElement(mergeCluster);
    }
}

```

Figure 5. Initial Version of AHC Algorithm

possible cluster pairs and merge the pair with its maximal similarity into one cluster. In its variant, we will consider allow more than one pair of clusters to be merged.

C. AHC Variants

This section is concerned with the variants which are derived from the AHC algorithm which was covered in Section III-B. The difference from the traditional version of AHC algorithm is to adopt the similarity metric between string vectors for computing the similarity between clusters. In this section, we derive the three variants by merging cluster pairs by adopt the threshold-based selection, allowing merge of multiple pairs in each iteration, and merging clusters within the radius. In this research, we adopt the three variants of AHC algorithm for implementing the word clustering system. This section is intended to describe the three variants of AHC algorithm.

In Figure 6, we illustrate the process of clustering data items by the first variant of AHC algorithm. The clusters are initialized with singletons like the initial version which was described in Section III-B. All possible cluster pairs are generated, and the cluster pairs whose similarities are more than or equality to the similarity threshold are merged. If there is no pair which satisfies the condition, clustering data items is terminated. In this variant, the number of clusters is decreased variably in this variant.

We illustrate the process of clustering data items by the second variant of AHC algorithm in Figure 7 as a pseudo

```

List clusterDataNumericalVectorList(List<String> vectorList, double similarityThreshold){
    int itemSize = stringVectorList.size();

    if(itemSize <= desiredClusterNumber)
        return null;
    List<Cluster> clusterList = new List<>();
    //Initialize Clusters
    for(int i = 0; i < itemSize; i++){
        Cluster cc = new Cluster();
        StringVector dataItem = stringVectorList.getElement(i);
        cc.addDataItem(dataItem);
        clusterList.addElement(cc);
    }

    boolean similarityFlag = false;
    //Cluster Data items in the bottom up direction
    while(similarityFlag){
        int clusterListSize = clusterList.size();
        similarityFlag = false;
        for(int i = 0; i < clusterListSize; i++){
            Cluster c1 = clusterList.getElement(i);
            for(int j = 0; j < clusterListSize; j++){
                Cluster c2 = clusterList.getElement(j);
                double similarity = sim(c1, c2);
                if(similarity >= similarityThreshold){
                    Cluster mergeCluster = c1.merge(cmax2);
                    clusterList.updateElement(i, mergeCluster);
                    clusterList.deleteElement(j);
                    clusterListSize = clusterList.size();
                    similarityFlag = true;
                }
            }
        }
    }
}

```

Figure 6. String Vector based AHC Variant 1: Similarity Threshold Based AHC Algorithm

code. The number of cluster pairs which are merged into each iteration is given as an additional parameter in this variant. All possible cluster pairs are generated, and the similarity is computed for each pair. The cluster pairs are sorted by the descending order of the similarity, and the pairs within the rank are merged. The difference of this variant from the initial version is to merge multiple cluster pairs, instead of one pair.

```

List clusterDataNumericalVectorList(List<String> vectorList, double similarityThreshold){
    int itemSize = stringVectorList.size();

    if(itemSize <= desiredClusterNumber)
        return null;
    List<Cluster> clusterList = new List<>();
    //Initialize Clusters
    for(int i = 0; i < itemSize; i++){
        Cluster cc = new Cluster();
        StringVector dataItem = stringVectorList.getElement(i);
        cc.addDataItem(dataItem);
        clusterList.addElement(cc);
    }

    int clusterListSize = clusterList.size();

    //Cluster Data items in the bottom up direction
    while(clusterListSize > desiredClusterSize){
        List<Pair> pairList = new List<>();
        for(int i = 0; i < clusterListSize; i++){
            Cluster c1 = clusterList.getElement(i);
            for(int j = i; j < clusterListSize; j++){
                Cluster c2 = clusterList.getElement(j);
                Pair indivPair = new Pair(c1, c2);
                indivPair.computeSimilarity();
                pairList.addElement(indivPair);
            }
        }
        pairList.sortPairs();
        List<Pair> selectedPairList = pairList.getSubList(0, rank-1);
        clusterList.mergeClusters(selectedPairList);
        clusterListSize = clusterList.size();
    }
}

```

Figure 7. String Vector based AHC Variant 2: Merge of Multiple Pairs

In Figure 8, we illustrate the process of clustering data items by the last variant of AHC algorithm. The similarity threshold is given as an external parameter, and the number of other clusters whose similarity is greater than or equality to the similarity threshold is counted. In each iteration, the cluster with its maximal number of its similar clusters is appointed as the pivot, and the pivot cluster and its similar ones are merged into one cluster. This variant iterates selecting the pivot cluster in the current cluster list and merge the pivot and its similar ones, as the process of clustering data items. If there is no cluster with its similar cluster, the iteration is terminated.

```

List clusterDataNumericalVectorList(List stringVectorList, double similarityThreshold){
    int itemSize = stringVectorList.size();

    if(itemSize <= desiredClusterNumber)
        return null;
    List clusterList = new List();
    //Initialize Clusters
    for(int i = 0; i < itemSize; i++){
        Cluster cc = new Cluster();
        StringVector dataItem = stringVectorList.getElement(i);
        cc.addDataItem(dataItem);
        clusterList.addElement(cc);
    }
    //Cluster Data items in the bottom up direction
    while(true){
        int clusterListSize = clusterList.size();
        if(clusterListSize == 1)
            return;
        int maxSimilarClusterSize = 0;
        int maxIndex = 0;
        for(int i = 0; i < clusterListSize; i++){
            Cluster c1 = clusterList.getElement(i);
            int similarClusterSize = c1.countSimilarClusters(clusterList);
            if(similarClusterSize > maxSimilarClusterSize){
                maxSimilarClusterSize = similarClusterSize;
                maxIndex = i;
            }
        }
        if(maxSimilarClusterSize == 0)
            return;
        Cluster pivotCluster = clusterList.getElement(maxIndex);
        List similarClusterList = clusterList.getSimilarClusterList(pivotCluster);
        clusterList.mergeClusterList(similarClusterList);
    }
}

```

Figure 8. String Vector based AHC Variant 3: Radius based AHC Algorithm

Let us make some remarks on the three variants of AHC algorithm which are adopted as the approaches to the word clustering. In each iteration of the first variant, cluster pairs with their similarities which are greater than or equal to the threshold are merged. In each iteration of the second variant, a fixed number of cluster pairs with their highest similarities are merged. In each iteration of the third variant, a pivot cluster and its similar clusters are merged. We may derive more variants from the AHC algorithm by setting different policies on merging clusters.

D. System Architecture

This section is concerned with the architecture and the execution flow of the word clustering system. In Section III-C, we described the three AHC variants which are adopted for implementing the word clustering system. The initial version of AHC algorithm which is mentioned in Section III-B is replaced by its variants in the word clustering system which

was proposed. The process of executing the system is to encode words into string vectors and cluster them. This section is intended to describe the word clustering system which is implemented in this study.

The process of encoding the words which are given as clustering targets into string vectors. The offline clustering where all words are given at a time is assumed in implementing the word clustering system. The words are encoded into string vectors by the process which is described in Section III-A. In the system, the string vectors which represent the words are clustered by the AHC algorithms which are described in Section III-C. The online clustering will be considered where words are given as a continual stream in the next research.

The system architecture of the word clustering system is illustrated in Figure 9. The role of encoding module is to encode words into string vectors. The similarity computation module which is a nested module computes the similarity between numerical vectors by the process, which is described in Section III-A, they are clustered by one among the AHC variants which are described in Section III-C in the clustering module. Clusters of string vectors are generated in the clustering module, they are decoded into words, and word clusters are generated as the final output from the system. The upgrading point of the system architecture which is proposed in [11] is to replace the initial version of AHC algorithm by its variants.

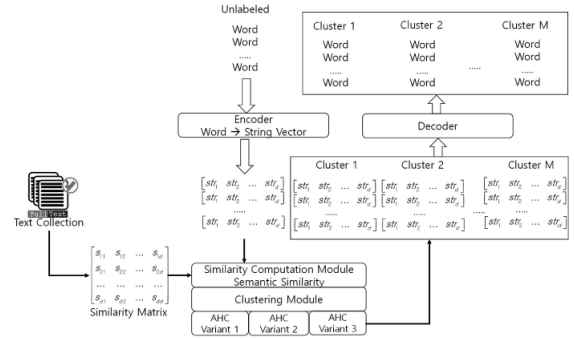


Figure 9. System Architecture

The execution flow of the word clustering system is illustrated in Figure 10. The words are encoded into string vectors by the word encoding process. The similarity metric between string vectors is used for computing the similarity between clusters. Data items are clustered by one which is selected among the three AHC variants which were described in Section III-C. The external parameter, similarity threshold, should be controlled in the first and the third variant for reaching the desired number of clusters.

Let us make some remarks on the proposed version of word clustering system which is presented in Figure 9 as its architecture. The offline clustering is assumed in implementing the system; this system should be expanded

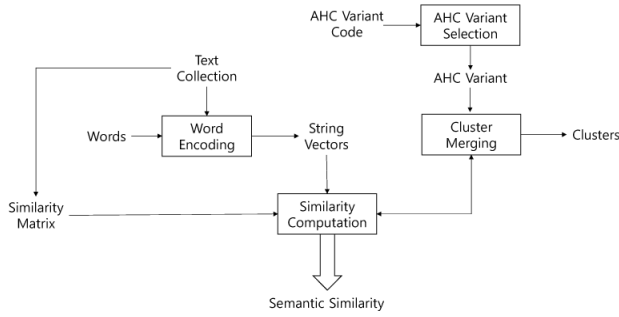


Figure 10. System Flow

into doing the online clustering in the next research. The upgrading point of this system is to replace the initial version of AHC algorithm by one of the three variants which are described in Section III-C. Words are encoded into string vectors, the similarity between clusters is computed by equation (4), and clustering data items proceeds by one of the three AHC variants. We expect the improvement of the clustering performance and speed by replacing the initial AHC algorithm by its variants.

IV. CONCLUSION

Let us mention the significances of this research as the conclusion. We derive the three variants from the standard version by allowing multiple clusters to be merged at a time. We encode words into string vectors and apply the similarity metric among them to the AHC variants as well as the standard version. We design the word clustering system by adopt the AHC variants with the proposed similarity metric. In the next research, we will implement the word clustering system as a real system.

Let us mention the remaining tasks as the further discussions on this research. It is necessary to improve the speed of computing the similarity between string vectors in the implementation level. Other clustering algorithms such as the divisive clustering algorithm and the online linear clustering algorithm should be modified into the string vector-based versions. The proposed versions of the AHC variants are applied to the text clustering as well as the word clustering. The word clustering system should be implemented by adopting the proposed approaches as a real program.

REFERENCES

- [1] T. Jo, "NTC (Neural Text Categorizer): Neural Network for Text Categorization", 83-96, International Journal of Information Studies, 2, 2, 2010.
- [2] T. Jo, "NTSO (Neural Text Self Organizer): A New Neural Network for Text Clustering", 31-43, Journal of Network Technology, 1, 1, 2010.
- [3] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview", 86-97, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2, 1, 2012.
- [4] K. Sasirekha and P. Baby. "Agglomerative hierarchical clustering algorithm-A Review", 83-85 International Journal of Scientific and Research Publications, 3,1, 2013.
- [5] K. Abainia, S. Ouamour, and H. Sayoud. "Neural Text Categorizer for topic identification of noisy Arabic Texts." 1-8, The Proceedings of IEEE/ACS 12th International Conference of Computer Systems and Applications, 2015.
- [6] Z. Nazari, D. Kang, M.R. Asharif, Y. Sung, and S. Ogawa, "A new hierarchical clustering algorithm", 148-152, The Proceedings of IEEE International Conference on Intelligent Informatics and Biomedical Sciences, 2015.
- [7] T. Jo, "String Vector based AHC as Approach to Word Clustering", 133-138, The Proceedings of 12th International Conference on Data Mining, 2016.
- [8] T. Jo, "Automatic Text Summarization using String Vector based K Nearest Neighbor", 6005-6016, Journal of Intelligent and Fuzzy Systems, 35, 2018.
- [9] T. Jo, "String Vector based AHC Algorithm for Word Clustering from News Articles", 83-86, The Proceedings of International Conference on Information and Knowledge Engineering, 2018.
- [10] P. P. Lakshmi and D. R. Rao, "Text classification using artificial neural networks", 603-606, International Journal of Engineering & Technology, 7, 1, 2018.
- [11] T. Jo, "Semantic String Operation for Specializing AHC Algorithm for Text Clustering", 1083-1100, Annals of Mathematics and Artificial Intelligence, 2019.
- [12] Taeho Jo, "String Vector based AHC Algorithm for Clustering Words Semantically", in Preparation, 2023.